A DATA ABSTRACTION MODEL

M.M. RIDWAN

and

Z. LOY

Technical Report 78/1

April, 1978

Department of Computer Science

University of Canterbury

Christchurch 1

NEW ZEALAND

ABSTRACT

An abstract model, called the data abstraction model, is proposed, and the basic DATAM constructs and representations are presented. The use of DATAM in the modelling and representation of the enterprise view of data is described.
The expressiveness of the data abstraction model is illustrated with examples and the capabilities of a DATAM model in subview formation and in evolution are considered.
DATAM provides a basis for the analysis of data models and analyses of the relational and network models are discussed.

Key words and phrases : enterprise view of data, concept abstraction, value abstraction, data abstraction model, database design, subview, evolution, entity-relationship model, relational model, network model.
CR Categories : 3.50, 3.70, 4.33, 4.34.

1. INTRODUCTION

The *enterprise view* of data is the manner in which the enterprise perceives its data resource.

The importance of a framework for the description of an enterprise view of data has lately been recognized (ANSI/X3/SPARC [1], Chen [7], Sibley and Kerschberg [16]). Such a framework provides the necessary interface for the definition of a process to produce a good database design.

Models for the representation of data fall into two types which will be referred to as abstract models and data models.

An *abstract model* is one which models the enterprise view with objects which are direct representations of distinct real world concepts.

A *data model* is one which provides structural representations for the instances of the associations between objects.

In the modelling of the real world, the concepts of the reality must be related to the objects of the database. The relationship is specified in a stated perception of concepts of the reality. This stated perception or abstract model (Biller and Neuhold [5]), provides the semantic reference with which one is to view the objects of the database.

The network and relational models (Date [10], Fry and Sibley [12]), among others, have been proposed as representations for the logical view of data. One inherent weakness of these data models stems from the fact that they do not stress the

enterprise view of data; that is, the abstract model or real world perception reference is not clearly defined. The approach of these data models has been to start from a representation and then define conditions and operations on the representation to reflect the dynamics of some real world concepts. In such a situation it is not clear what abstractions the model is meant to support.

An alternative approach is to create an abstract model by identifying and defining formalized data concepts and then proceeding to the representation, resulting in data descriptions that are in much better accord with the enterprise view. An example of this abstract model approach is Chen's entity-relationship model [6] which begins with a definition of formalized entity and relationship concepts for the real world and then considers representations of these conceptual objects.

In this paper we present a generalized data abstraction model which is introduced in Section 2. DATAM is based on definitions of formalized entity, relationship and event concepts. Although these concepts are different from those of Chen's, the data abstraction model can be viewed as a generalization or extension of the entity-relationship model. Two examples in Section 2 illustrate the capability of DATAM in modelling semantic information of the real world.

The abstraction of values for the representation and storage of actual instances of the formalized concepts is considered in Section 3. Section 4 discusses consistent

subviewing of the abstract model allowing for the perception of selected subsets of the enterprise view.

In general, the enterprise perception of the world changes over time. To continue as an accurate representation of this perception, the abstract model itself needs to evolve appropriately. Section 5 discusses operations for the evolution of DATAM models.

Besides being a framework for data description and representation, DATAM can also be used in the analysis of data models. A DATAM interpretation and analysis of the relational and network models is presented in Section 6.

2. THE DATA ABSTRACTION MODEL (DATAM)

2.1 The Modelling Process

In discussing abstractions, it is necessary to distinguish between the universe of real world objects and the universe of database objects. The modelling of the real world in database terms centers around the modelling of the existence and description of objects of the real world and the association between these objects. The descriptions and relationships of objects in the real world vary in type and extent with the state of the perceiver. However, database objects have to be unequivocally described and have well defined associations at every instant. Abstractions serve this purpose by providing constructs with which the database universe is to be defined. The formulation of different semantic constructs arises from the perception of different types of associations and descriptions of objects in the real world.

In the modelling of data, multiple levels of views of data or realms of interest (Biller and Neuhold [5], ANSI/X3/SPARC [1]) can be identified. The upper level concerns ideas about the real world and the lower level deals with the representation or recording of these ideas.

In the data abstraction model, basic formalized constructs for the model are first formulated from a perception and crystallization of real world concepts. The representation of and operations on these constructs are then

defined. This collection of formalized constructs, information structure and operations constitutes the data abstraction model presented here.

Ideas about the real world can be separated into two areas - one concerning concept types and the other concerning the instance values of these concept types. These two areas will be referred to here as the concept abstraction and value abstraction domains respectively. The basic diagrammatic representations used in DATAM are presented in Figure 1 which shows the representations for the formalized constructs at the concept abstraction domain and the representations of value abstractions. The formalized constructs or concept abstraction types are described in this Section which also presents two modelling examples while value abstractions are discussed in Section 3.

2.2 Abstractions of the Reality

An abstract model is a "set of all abstract states which are abstractions of possible real world states" (Biller and Neuhold [5]). In the data abstraction model we identify the following (concept) abstraction types or database constructs.

1. *Entity* which models the existence of a real world object and its capability of being described.
2. *Attribute* which represents the concept of the description of an object where the descriptor object is not itself described.
3. *Relationship* which represents the concept of the mutual description or association between two entity or event types.

4. *Event* which models a describable object in which the object represents an association between two or more entity or event types.

Diagrammatic representations of these abstraction types and their associational connectives are given in Figure 1(a). Each of the abstraction types are considered in detail in this Section.

In this paper, the adjective *database*, or the prefix db- may be used to qualify an abstraction type (e.g. database entity, db-attribute) to indicate that the abstraction type of the model is meant. Generally, when no qualification is stated, the implied reference to the model or real world concepts will be clear from the context.

2.2.1 Entity. The most basic abstraction is that of the *entity* which represents a real world entity. A real world entity (ANSI/X3/SPARC [1]) is "a person, place, thing, concept or event, real or abstract, of interest to the enterprise". Within the database universe, an entity is perceived as a single, uniquely identifiable object. A real world entity set is a collection of real world entities of the same type. In the abstract model, this is reflected by a database entity set which is a collection of database entity instances. Figure 2 illustrates a simple DATAM diagram for a "supply" model. The db-entity abstraction types shown for Project, Supplier and City represent the corresponding real world entity types of interest.

2.2.2 Attribute. Associated with an entity may be

several *attributes*, representing the descriptions (properties, characteristics) of the real world object which the database entity is representing. In the real world an object perceived as a property (of another object) may also be viewed as an entity in its own right. An example is *colour* which may be used to describe an entity, car; but colour itself may be an entity in that it may be described by, for example, wavelength. This multi-context use of the same object is always possible in the real world. However, the contexts themselves are well-defined. An object is a property when it is being used to describe another object; it is an entity when it itself is being described. Thus in the database abstractions, a database entity is described by db-attributes or, equivalently, db-attributes describe a db-entity. Thus the role of a database object as attribute or entity in DATAM has a fixed context.

In Figure 2, Project is seen as an entity while Special-Equipment and Schedule are seen as attributes as they are not themselves described. The Project : Special-Equipment association, for example, represents the description "requires Special-Equipment" and indicates the association between instances of the Project entity to instances of the Special-Equipment attribute. This mapping or association between Project, J, and Special-Equipment, SE, is shown as *m:n*. For the model, this instance association can be expressed as $J \overset{m}{:} \overset{n}{} SE$, where the notation, $\overset{m}{:}\overset{n}{}$, will be used to denote associational concepts.

2.2.3 Relationship. With the entity and attribute abstraction types, a database object can then either be described or be a descriptor. In the real world,situations often arise where an object is both described as well as being a descriptor. The *relationship* abstraction type is introduced to reflect this associational concept. The description of an entity by another entity is differentiated from the description by an attribute (see also Schmid and Swenson [15], Tsichritzis and Lochovsky [19]). The former is called a database relationship. For example, a person may be described by his *relationship* ("owns car") with a car which is itself perceived as an entity; while the description "has hair colour" is an attribute, as hair colour is not viewed as an entity here. Note that both attributes and relationships model the *concept* of describing. They are description types and correspond (in English) to verb phrases - e.g. "has colour", "owns car", "can be contacted by phone number", "is the spouse of". However, a relationship facilitates an association between two entities, while the association between an entity and its attribute is direct. Thus there may be many relationships between any two entities, that is, the entities may describe each other in many ways, while an attribute represents only a single conceptual description of a particular entity.

Figure 2 shows two relationships between Supplier, S, and City, C, illustrating a multiple relationship situation. S and C describe each other through the relationships. For

example the $n : 1$ relationship, R_1, can be viewed as "Suppliers S have major locations in Cities C" or the inverse "Cities C are the locations for Suppliers S". The other $(m : n)$ relationship, R_2, between S and C indicates "auxiliary location".

2.2.4 Event. The *event* abstraction type represents the real world viewing of a description type as a describable object. For example, the description type "owns car" (person-entity : car-entity) may be viewed as the (event) object "ownership of car" which might be described by, e.g. "date of ownership", and "conditions of ownership". The event "ownership of car" is the noun dual of the verb phrase "owns car" (Biller and Neuhold [4]).

The perception of a describable event implies cognizance of the entities and the relationship from which the event is derived.

An example of an event is shown in Figure 2. The event, E, Project-Supply, involves the entities Project and Supplier and represents the perception of the $(m:n)$ relationship between Project J and Supplier S as a describable object. The event implicitly contains the relationship between J and S. The event, E, is described by an attribute, Part, representing the Part supplied in the supply to J by S.

The existence of event instances in the database does not preclude the possible independent representation for the corresponding database relationship. For example, separate instance representations may exist in the database for both

the event, E, of Figure 2 and the J:S relationship. In such a situation, both representations must develop consistently. To avoid this potential integrity problem (and also the inherent conceptual redundancy) the rule is made that where the database event is defined then the implied assocations between the objects involved in the event is only available through the event and is not independently defined as well. That is, the specification of a database event contains the specification of its corresponding relationships.

2.2.5 n-way event. An n-way event may be composed from n ($n \geq 2$) objects. For example, Figure 3(a) shows a 4-way event representing an association between the entities Project, Supplier, Part and Quantity. This event implies the existence of the corresponding 4-way relationship between the four entities.

However, it should be noted that any (other) explicitly defined relationship between the objects of an event does not imply any associations with the event itself. For example, in Figure 3(a), if a relationship, R_1, is explicitly defined between J and S, then no association can be implied between this relationship and the event, E_1.

The event of Figure 3(a) may be differentiated representationally into either Figure 3(b) or 3(c), among other possible diagrams. In Figure 3(b), the event E_2 represents the supply of Parts P by Suppliers S. This event, together with the entity, Project, J, constitutes

event E_3 representing the supply of P by S to J. (The membership of E_2 in the event E_3 is indicated by the arrow). The event E_3 has a relationship with the Quantity, Q. Figure 3(c) represents an event, E_4, of J-S-P which has a relationship with Q.

Each of the diagrams, Figures 3(a), (b) and (c) is considered to contain the 4-way relationship between the entities, J, S, P and Q. As they are shown, these three diagrams are equivalent conceptually over the totality of instances. Differences, however, become apparent when descriptions for a particular relationship are included. It may be required, for example, to describe the relationship "the supply of P by S" by "regularity of supply". In this case, an association between the event E_2 of Figure 3(b) and an attribute "Regularity of Supply" is formed. No other diagrams allow directly for this description. Thus, in the modelling process, Figure 3(a) may be used when there are no descriptions for the embedded relationships. When such descriptions exists the appropriate model (e.g. Figure 3(b), 3(c)) has to be chosen. In the case where such a description is perceived and included later, Figure 3(a) may be evolved to the appropriate model to reflect the change in view. (The evolution of data models is considered in Section 5). It should be stressed that while the different models of Figures 3(a), (b) and (c) contains the same 4-way relationship between the entities, each represents a different perception. The choice of model is determined by the enterprise view and its anticipated development.

2.2.6 Relationship and event definitions. Figure 3 also illustrates the reasons for the entity and event definitions made in Section 2.2.

As discussed above (Section 2.2.5), equivalent diagrams involving events and relationships can be constructed for a given number, n, of entities. All these models contain the direct n-way relationship between the entities. For example, Figures 3(a), (b) and (c) contain the 4-way relationship between the entities J,S,P and Q.

The point to note is that the models in Figures 3(b) and 3(c) have relationships between only two objects. Since any n-way relationship can be perceived to be contained in a model which has events together with relationships between only two objects, a construct for the direct representation of a relationship of more than two objects is unnecessary. Because of this, and with economy of concept in mind, a relationship is defined,therefore,to exist between only two database describable objects (entities or events).

The definition of an event in Section 2.2 allows only entities or events to be components of it and precludes attributes. As an elucidation of this point, consider the case, for example, of the event of Figure 3(a) but with Supplier and Part being attributes instead of being entities. Then a later possible change to Figure 3(b) results in an event E_2 representing an untenable attribute-attribute (S:P) association. To overcome this representation inconsistency

the definition for a db-event allows only describable objects (entities or events) to be members of it. It should be noted that although attributes cannot be components of an event, they can still be used to describe an event (see, for example, Figure 2).

2.3 Illustrative Examples

To illustrate the concepts discussed, two examples of the modelling process will be given here. For the first example, an analysis of the relational, entity-relationship and DATAM model approaches is presented. The examples demonstrate the expressiveness of DATAM in the modelling of the real world.

2.3.1. Example 1. The following example, from Date [10], is considered :

The enterprise view concerns Teachers (T), Subjects (J) and Students (S) and their associations. The perceived associations are :

1. For each Subject, each Student of that Subject is taught by only one Teacher.
2. Each Teacher reaches only one Subject.
3. Each Subject is taught by several Teachers.

In terms of functional dependencies, the above statements may be expressed as :

$$S, J \rightarrow T$$

$$T \rightarrow J$$

In the relational model, the representation of the above situation by the relation R (S,J,T) leads to processing

anomalies (Date [10]). Date gives a normalization solution which involves decomposing the relation R (S,J,T) into the relations

R1 (S,T) and

R2 (T,J).

In this case, however, the relations R1 and R2 now require some form of interrelation operators to maintain the semantics which is not indicated by the schema itself. In particular, conditions 2 and 3 above are represented by relation R2, but condition 1 is lost in the schema and becomes a user responsibility. For example, let the instances of R1 and R2 be as shown in Figure 4. The instances satisfy all conditions. However, in the addition of tuple (S1,T2) to the relation R1, condition 1 (S,J→T) is violated. This violation is discernable only through validation with respect to relation R2 and is not obvious from the relational schema.

Using the entity-relationship diagram (Chen [6]), a model for the given example is shown in Figure 5. The given conditions 2 and 3 are represented by the ER-relationship R3 and condition 1 is represented by the ER-relationship R4. (The prefix ER- is used to distinguish concepts in the entity-relationship model from the concepts presented in this paper. Note also that the relationships R3 and R4 in Figure 5 do not represent DATAM events). Although the entity-relationship model represents the situation better, there is still a semantic aspect in the example that it does not encompass. Let the instances for the ER-relationships be as shown in Figure 5. These instances satisfy the

mapping conditions of the entity-relationship diagram. However, the first and fourth instances of R4 reveal an implied violation. An implied condition or consequence of condition 1 is that the Teacher for the Student of a Subject is the Teacher of that Subject. The instances, (J1,S1,T1) and (J2,S3,T1), imply that T1 teaches both Subjects J1 and J2, violating condition 2. The entity-relationship diagram thus does not fully model the given situation.

In DATAM, the semantic constraint in condition 1, that the Teacher involved is the Teacher of that Subject, can be specified explicitly as part of the model. This is possible because DATAM distinguishes between events and relationships, and allows relationships to be specified between two events. The DATAM model for the example is shown in Figure 6. The event E_1 represents the event that students take subjects, and event E_2 represents the event that teachers teach subjects in a manner (*n:1* mapping) satisfying conditions 2 and 3. The database relationship R_5 represents the relationship between events E_1 and E_2; that is, for each subject, each student taking that subject (event E_1) is taught by only one teacher of that subject (event E_2). R_5 explicitly satisfies condition 1. Instances for the model are also given in Figure 6.

2.3.2 Example 2. This example involves a supply (Project-Supplier-Part) model. The perceived associations are :

1. A Supplier (S) supplies a Project (J) with a Part (P) in at most one Quantity (Q).
2. Each Supplier will only supply Parts in particular Quantities; therefore, the Supplier in (1) is chosen from a set of Suppliers which supply that Quantity for that Part.

In terms of value associations, these can be expressed as:

1. S,J,P $\overset{n}{:}\overset{1}{}$ Q
2. Q,P $\overset{m}{:}\overset{n}{}$ S

Note that the notation, $\overset{m}{:}\overset{n}{}$, indicates value (instance) associations. For example, the second condition states that the set of allowable Suppliers, S, is determined by the Quantity of the Part required; or, equivalently that the Quantities of Parts that can be supplied is determined by the Supplier (with a $m:n$ association of instances).

As with the first example, it is difficult to express this situation accurately with most models. (In particular it leads to a non-BCNF schema (Bernstein [3]) in the relational model).

The DATAM diagram for this situation is given in Figure 7. Event E_1 represents the event of Suppliers, S, supplying Project, J with Parts, P. E_2 represents the particular Quantities, Q of Parts. Event E_3 represents the supply of particular Quantities of Parts by a Supplier (condition 2). The database relationship R_1 represents the relationship, $E_1 : E_3$ (strictly, $E_1 \overset{n}{:}\overset{1}{} E_3$), that the supply of Parts to a Project by a Supplier (Event E_1) is of the correct Quantity (thus satisfying condition 1).

The above examples illustrate the capability of DATAM in the modelling of data. Situations involving multiple associational criteria, such as in the examples, often occur in perceptions of the real world. Such situations are modelled straightforwardly in DATAM. The expressiveness of DATAM facilitates accurate modelling of the real world. Constraints can be included in the model rather than be delegated as an external responsibility, so that the database models the perceived enterprise view more closely.

3. VALUE ABSTRACTION

3.1 The Instances of Abstractions.

The modelling of the real world is the specification or the fitting of real world concepts as abstraction types using the available abstractions (as defined, for example, in Section 2 for DATAM). The defined abstraction types and the associations among them represent the agreed upon model of the real world concept types of interest to the enterprise. The *instances* of these real world concepts are represented by the value instances of the abstraction types and their associations. Concepts for the representation of value abstractions are presented in this Section. These include the instance set, range set, tokens and identifiers. The diagrammatic representations for these concepts is given in Figure 1(b).

The value abstraction concepts presented allow the dependent-entity situation (see Section 3.2) to be represented without difficulty. The representation of dependent objects, (entities and events), is discussed in Section 3.2 while the construction of a suitable data definition language for DATAM is considered in Section 3.3.

3.1.1 Instance Set. The association between the database objects is represented finally by the association between values for these objects. Thus an entity-attribute association, for example, is represented by the association of values from two sets - one representing the instances of the entity type and the other the set of attribute-instances. For each abstraction type, then, there exists a set of values which participate in the association instances to represent

the values the real world is perceived to have at that given instant of time. The sets viewed in this form will be called *instance-sets*. It should be noted that these sets of instance values may contain non-unique values.

3.1.2 Range Set. Each unique value from the instance sets for an abstraction type is a member of a set of values. This set then represents the allowable values that the corresponding real world concept may have at any time. Such a set will be called a (value) *range set*. These range sets are therefore the abstractions of the instances that real world concepts may have. Figure 8 shows the range sets for the attributes of Home-address and Office-address. These attributes describe the entity, Person. The entity, Phone, has an attribute, Address-of-Phone, and the range set corresponding to this attribute is also shown. While a range set represents the concept of value abstraction, the actual values that the database object has at any given instant is represented by an instance set.

3.1.3 Super-range set. There may be situations where a range set encompasses or contains another range set in that it represents a more general concept than the other. It is sometimes useful to refer to such a set separately as a *super-range set*. In Figure 8 the range set for the attribute Address-of-Phone is shown. For the enterprise (telephone company) this is perceived to encompass the Person's (subscriber) Home and Office addresses. "Address-of-Phone"

is thus a super-range set and the conceptual containment of the other two range sets within it is represented explicitly by the solid lines joining the Home-address and Office-address range sets to the Address-of-Phone range set.

3.1.4 Token. The instance sets corresponding to each of the functionally different concept abstractions (that of being described and that of describing) are treated separately because of the different semantic interpretations. Identical values in instance sets of entities and events represent the *same* real world object, while identical values in instance sets of attributes represent the *same description* of different objects. In the instance sets of describable db -objects each unique value represents a unique object in the real world. These values will be called *tokens*.

In Figure 8 the range sets for the entities Person and Phone are token range sets as each value represents a unique describable object. As events are describable objets, they also have token range sets in the value abstraction domain. Thus, for example, each instance of the events E_1 and E_2 in Figure 6 has a corresponding token.

3.1.5 Identifiers. Often it is possible to identify a unique object in terms of its descriptions. For example, a person may be identified uniquely by the person's name, weight and height if it is perceived that no two persons have the same values for all of these descriptors. Such a combination of values of associated attribute instance sets used to identify an entity (or event) is termed an *identifier*.

This is illustrated in Figure 8 where the identifier set {name, weight, height} is considered equivalent to the token set, I.D.#, and this equivalence is indicated by the dotted line.

It should be noted that the concept of identifiers is different from that of tokens, since an identifier merely identifies an object while a token is the object. To illustrate this, consider the example in Figure 8 where the entity-type Person, with token range-set I.D.#, has the attribute, Home-address, and it is perceived that Home-address is an identifier. Now if the Home-address of a Person with I.D., #x, say, is deleted (representing, for example, that the person has moved but has not yet found a residence), then although it is still true that a Home-address, where it exists, uniquely identifies a Person, it is now not possible to identify Person #x through Home-address, since this Person's Home-address does not exist. The existence of the Person in the database is reflected by the token, I.D.#, and not by Home-address. If the actual person leaves the perception of the database, then its token is deleted together with all its attribute associations. To further illustrate the difference in concept, it may be that at a later time it is to be perceived that different Persons may have the same Home-address. In this case, Home-address is no longer an identifier of the entity type Person.

Since the model should evolve with the perception of the real world, such changes of attributes to identifiers (and vice versa) are permitted.

3.2 The Representation of Dependent Objects

It has been stated that the existence of a describable object is represented by a token and the object may sometimes also be identified by its attributes. A token range set can always be defined for any describable object type (entity or event). Situations arise, however, where the existence of an object is determined by the existence of another object.

For example, in a corporate database (Figure 9(a)), an entity, Dependent-of-Employee, may be perceived only in the context of the Employee entity on which it is dependent (Chen [6]). Thus the deletion of an Employee from the database requires the deletion of all the Employee's dependents as well as their attributes. Also, the identification of a Dependent requires, and is to be only possible through, the identification of the related Employee. In terms of processing this means that the processing of Dependents is only to be done within the processing of Employees. Figure 9(a) illustrates this situation, where the entity, Dependent-of-Employee, is drawn with dashed lines. This entity has no tokens, as implied by the dependent identification, since the existence of tokens allows direct identification of the object. Instead, the identifier for this Dependent-of-Employee entity is a concatenation of the token of the entity (Employee) on which it is dependent together with appropriate attributes of the dependent-entity itself. This identifier then allows the dependent-entity

to be uniquely identified. Figure 9(a) shows the identifier for the Dependent-of-Employee entity being made up from the token for the Employee entity and the dependent's Name attribute. The relationship, R_d, between the Dependent-of-Employee entity and Employee entity (through which the former is dependent on the latter) is called a dependency-relationship.

From the above example, a *dependent entity* can be defined as an entity which does not have a token and its identifier contains a token of the entity on which it is dependent through a relationship. Note that the owning entity may itself be a dependent entity. In this case the identifiers of all the nested dependent entities relate to the token of the final owning entity.

A *dependency-relationship* is the relationship through which one entity is dependent on another. If this dependency-relationship is itself described, then the resultant event is a *dependency-event*. Figure 9(b) shows the dependency-event, E_d, obtained when the dependency-relationship, R_d, of Figure 9(a) is perceived as a describable object.

A dependent object may also be a component of a regular event. For example, in Figure 9(a) the dependent entity, Dependent-of- Employee, together with the entity, Company-car, form the regular event, Usage-of-Company-Cars.

Any db-event that has a dependent object as a member does not have a token. Its identifier consists of the identifier of the dependent object and the tokens of the other member objects. For example, the identifier of the dependency-event E_d in Figure 9(b) consists of the token

of the owning entity (Employee) and the identifier of the dependent entity (Dependent-of-Employee). Since the token of the owning entity is already part of the identifier of the dependent-entity, the identifier of the dependency-event therefore has the same range set as that of the dependent-entity (although their instance sets are necessarily different).

3.3 Data Definition Language

During the logical design of the database, the objects and their associations have to be defined and declared to the database management system. These declarations are typically stated using a specified data definition language (Date [10]). The declarations of the DATAM concepts presented here are straightforward and the construction of an appropriate data definition language presents no difficulty and will not be discussed in detail here. As examples, declarations for the abstraction types could be of the following form:-

Entity Person
    Token is ID-number
    Identified by <Name, Weight, Height>
    Attributes are <Home-address, Office-address,
        Name, Weight, Height>

Event Project-Supply
    Token is PS-number
    Members are <Project, Supplier>
    Attribute is <Part>

Relationship Major-Location
    Members are <Supplier, City>

The first declaration refers to Figure 8 while the other two are for Figure 2. (The declaration for the composite-attribute, Date-of-Birth, for Figure 8 is discussed in Section 5.1.5).

It should be noted that the data definition language corresponds directly to the DATAM abstraction types and reflect directly the enterprise view of data. This should be contrasted with the declaration of some models, e.g. the relational model, where the declarations are in terms of representational structures rather than with respect to specific abstractions of the real world.

4. SUBVIEWS OF THE ABSTRACT MODEL

4.1 Subviews of DATAM models

Subviewing is defined as the viewing of a subset of the modelled world. This facility is important to the enterprise as groups and individuals within it may want or be allowed to see only a subview of the total view. Whilst the term subview relates to the enterprise view, it should be noted that the terms subschema and submodel (Date [10]), though similar in concept, generally relate more to representational structures.

In the data abstraction model, each construct is based on a well defined associational concept of the real world. Hence, the forming of subviews is determined directly by the meaningfulness of the action on the corresponding concepts of the real world. Thus well-defined rules can be formulated for the construction of subviews. The rules for subviewing in DATAM will be described in this Section. These rules ensure the formation of a subview that represents a valid and consistent perception of a subset of the enterprise view.

4.1.1 Subviews with Entities. The most basic abstraction is the database entity which represents the existence of an object in the real world. Since it is meaningful merely to perceive the existence of an entity, i.e. without its descriptors it is therefore meaningful for a database entity to exist independently of all other abstractions. Thus, it is consistent to form a subview of Figure 2

consisting of any number of the db-entities, e.g. Project and City. There is no logical necessity that these abstraction types be associated in the subview.

4.1.2 Subviews with Attributes. A database attribute represents a description of a real world object. This means that its existence is meaningful only in association with the database entity or event which the attribute describes. Also, it is possible to perceive only some of the descriptions of an entity. Thus, for the DATAM diagram of Figure 2, a valid subview with the attribute Schedule is one containing this attribute and the entity, Project.

4.1.3 Subviews with Relationships. Database relationships represent the mutual description between objects, so that in a model the existence of a relationship is meaningful only in terms of the entities or events which it relates. Thus a subview with the relationship R_1 of Figure 2 requires the inclusion of the entities Supplier and City.

4.1.4 Subviews with Events. A database event requires the explicit perception of the objects involved in the association which the event represents as a describable object. Thus a minimum valid subview containing the event E in Figure 2 requires the concurrent perception of the entities Project and Supplier.

Note that the minimum valid subview containing the relationship R_5 in Figure 6 is the complete diagram itself, as the perception of the relationship R_5 requires the

perception of the events E_1 and E_2 which in turn require the perception of the entities, J, S and T.

5. EVOLUTION OF THE ABSTRACT MODEL

5.1 Evolution of DATAM models

The construction of an abstract model represents the perception of the real world at the time of construction. As perceptions of the real world change with time, changes must also be made to the abstract model to reflect these changes in perception, in order to maintain the usefulness of the database (Swartout et al. [17], Navathe and Fry [14], Chen [7]). Such changes to the abstract model result in either an increase or decrease in the number of concepts modelled; and the evolution process is correspondingly defined as *progression* or *regression*.

This Section describes the basic operations that are necessary for the evolution of the abstract model to reflect the perceived changes. The difference in this approach to some others (see Biller and Neuhold [5], Kerschberg et al. [13]) is based primarily on the differences in the abstraction process. In particular, in DATAM, database attributes are viewed as being in the concept abstraction domain and represent an object-type construct rather than a value range. This emphasizes that attributes are strongly perceived in the enterprise model instead of the more usual viewing of attributes as mappings from an entity set to a value set. Thus database attributes and entities are seen to be functionally similar in that both have instance sets which are defined on value ranges, such that an entity-attribute

association is represented by the association of the values in their respective instance sets.

In this Section, operations for the evolution of DATAM models is discussed. As regression is the inverse of progression, for each sequence of progression operations, there is a sequence of regression operations to reverse the operation. As such, the following discussion will be limited to the factors involved in progression operations. Also, the straightforward operations involving the entry and exit of objects from the database, such as the addition of a new entity type, are not considered here. Sections 5.1.1 to 5.1.4 discuss operations on the transformation of one abstraction type to another while Sections 5.1.5 and 5.1.6 discuss operations defined on the instance sets of abstraction types and introduces the concepts of composite-attribute and sub-object types. The effect of the evolution process on subviews is discussed in Section 5.2.

5.1.1 Attribute→Entity. A db-attribute represents the description of an entity. If the descriptor (attribute) itself becomes perceived as a db-entity, then the previous entity-attribute association becomes a relationship, representing the mutual description between the entities. This change in perception has repercussions on the underlying range sets as the change of an attribute to an entity means that its value range set becomes a token range set. An example is given in Figure 10. When the attribute, Address-of-Phone, becomes an entity (with its own attribute,

Category-of-Address), the Phone : Address-of-Phone association becomes a relationship, R_2, and the Address-of-Phone value range set becomes a token range set.

Such a change may also force the change of other attributes into entities. This would occur if the attribute being changed represents a description which conceptually encompasses others in the database. This containment is evident in the associations among value ranges (and is shown explicitly in the value abstraction domain of the DATAM diagram). Hence where the value range of the attribute being changed is a super-range set, then the contained ranges also become token ranges, which thus changes the attributes defined on them into entities and their associations into relationships. In Figure 10, for example, the change in perception of the attribute Address-of-Phone into an entity forces a change in the other two attributes (Office-address and Home-address), which are conceptually encompassed by the Address-of-Phone attribute, into entities.

A similar change also occurs when a sub-range set becomes a token set as this implies that its super-range set has also to be a token set. For example, the evolution of the attribute Home-address in Figure 10 into an entity would also result in the change of the attributes, Address-of-Phone and Office-address, into entities.

5.1.2 Entity→Event. An event (object) of the real world may be modelled as a db-entity. For example, in Figure 10 the real world perception of a "marriage" event is modelled as a db-entity (Marriage). If the objects involved in the real world event are themselves modelled and their associations with the db-entity which represents the event is to be indicated, then this db-entity is changed to a db-event. This progression is illustrated in Figure 10 where the couple involved in the marriage is to be indicated. Note that an instance of the Marriage event involves two separate instances of the Person entity.

5.1.3 Relationship→Event. A database relationship represents the association or mutual description of two db-entities or events. If this relationship is to be viewed as an object or is to be described, then it is modelled as an event. This change is shown in Figure 10 where relationship R_1 becomes event E. This operation requires the creation of a token range set for the resultant event. As the associated values from the token instance sets of the objects involved in the event are always identifiers of all instances of the event, a token range set of the event can therefore be defined as the concatenation of the values of the token range sets of the objects involved. For example, Figure 10 shows the token range set for the event E as being made up from the token range sets of the entities Person and Skill. Thus, where the event token is to be defined by the member objects, then no explicit

declaration of the token need be made in the data definition language (see Section 3.3).

A similar change occurs when an entity-attribute association is to be described. The attribute is first changed to an entity (Section 5.1.1) and the resulting relationship then changed to an event.

The progression of a dependency-relationship to a dependency-event is also similar and is discussed in Section 3.2.

5.1.4 Dependent Entity→Regular Entity. A dependent entity is one whose existence and identification is possible only through its owning entity with respect to a corresponding relationship. The change of a dependent entity to a regular one means that the entity is to exist and be identifiable independently of other objects. This thus simply involves the creation of a token range set for the dependent entity and to indicate the corresponding change of the dependency-relationship to a regular one. For example, in Figure 9(a),a perceived change of the dependent entity, Dependent-of-Employee, to a regular one would be effected by the creation of a token range set for this entity. The dependency-relationship, R_d, in Figure 9(a) then becomes a regular relationship.

5.1.5 Composite-attribute. The above Sections discussed operations on abstraction types. This Section and the next discusses operations that bear on instance-based constructs and thus affect the value associations of the instance sets.

One concept which in principle is commonly available is where a descriptor instance of an object is the physical concatenation of other descriptor instances. Consider, for example, the following declarations (in a data definition language) :

Attribute Date-of-Birth

Members are <Day, Month, Year >

Attribute Person-Name

Members are <First-name, Last-name>

where the attribute, Date-of-Birth is seen to consist of the attributes Day, Month and Year, and the attributes First-name and Last-name together constitute Person-Name. The motivation behind such a decomposition is typically the requirement to be able to refer to, and to manipulate portions of,a descriptor instance independently, as well as to refer to it as a unit.

The actual operations of the fragmentation and concatenation to represent the above concept are based on the instances of the attributes. It is the attribute value instances that are composed or decomposed, with the value range sets serving, as usual, only to define the scope of the constituent values.

To reflect this situation, the *composite-attribute* construct is specified in the concept abstraction domain as an association among attributes, and an example for Date-of-Birth is shown in Figure 8. Note that the diagram for the composite-attribute, Date-of-Birth, also denotes the ordering, left to right, that is implied in the definition of the physical concatenation of its components.

The operationsnecessary to effect changes to the composite-attribute are operations for the addition of attributes to and deletion of attributes from a composite-attribute. These operations conceptually include operations for the composition of a composite-attribute from a set of attributes and the corresponding decomposition.

It should be noted that the above definition of the composite-attribute allows for the construction of nested attributes.

5.1.6 Sub-entity and Sub-event Types. Given an entity type, it is possible to derive a subset of its instance set dynamically, using appropriate access operators. Such a subset can then be used in subsequent processing. This conceptual subset of the entity is not,however,permanent as it is not perceived as a separate object-type in the model, and would therefore not be automatically maintained across sessions. To make the perceived subset of the entity permanent a corresponding entity-type has to be created, the tokens of which are necessarily contained in the token range set of the source entity-type. The object thus created is a *sub-entity-type*. For each token of the sub-entity there is a corresponding token of the source entity such that they both represent the same real world object. Thus, any operation on the token of a sub-entity is applicable to that token in the source entity, while for the reverse operation, this is true only for those tokens that also exist for the sub-entity.

To illustrate this, consider the example in Figure 11, where Son is a sub-entity of Child and the containment of the Son token range in the Child token range is indicated by the arrow. The perception of a new Son, indicated by the creation of a new unique instance for the Son sub-entity type, would trigger the entering of this new Son instance into the instance set of the Child entity (if it is not already there). The reverse case,however, is not immediate. To include a new Child instance into the Son instance set requires that the Child be male. This condition, however, cannot be established if sex is not modelled as an attribute of Child. Thus, an external decision is required in such a case.

The instance association between a sub-entity and source-entity types are therefore not symmetric. The inclusion from sub to source is immediate. The reverse is conditional, where the inclusion may be system controlled if the database models enough concepts to define the inclusion criteria.

A similar process can be described for the event abstraction type.

Associated with this concept would be operations for the forming of a sub-object-type from an existing object-type either with respect to external conditions or system controlled criteria. The reverse operation forms an object-type which contains objects already existing. (Note that these operations are similar to corresponding

operations of Chen [7] for the entity-relationship model and that of Swartout et al. [17] for network databases).

5.2 Evolution and Subviewing

Since subviews represent consistent subsets of the enterprise view, they can be constructed by the process of regression.

The implementation of a subview has a bearing on the actions required to maintain consistency in the event of an evolution (particularly, a regression) of the enterprise view. Since a subview is defined with respect to a specific model, its consistency is determined by the state of the model at the time of the subview construction. However, subsequent evolutions may affect the consistency of the subview representation as a subset of the model. That is, the subview may contain associations or objects not now represented by the database.

For example, if the construction of a subview results in a complete sub-database embodying all the object and association types, instances and value range specification, then any regression of the database would require a separate parallel action on the subview. On the other hand, if the subview is implemented such that it shares the instances of the database, then a change in the instances of the database in effect also changes the appropriate subview instances.

The choice of the level of definition of subviews is determined by the degree of awareness required by the user. It is usually sufficient to adopt the second approach discussed above, and construct subviews as subsets of the concept abstractions with sharing of the instances.

6. ANALYSIS OF DATA MODELS

6.1 DATAM as a Semantic Reference

Since data models are based on representation structures they are subject to interpretation as it is not clear what concepts are represented. In the analysis of data models, an interpretation can be fixed in terms of an external semantic reference. Abstract models can provide such a reference. In general, an interpretation of a data model imposes a discipline on the construction of its structures. The use of DATAM in the analysis of data models is presented in this Section.

As data model structures are representations of instances of associational concepts, correspondence with DATAM is best seen through comparisons with DATAM instance representations. Section 6.2 describes the representations that will be used. Analyses of the relational and network models are presented in Sections 6.3 and 6.4 respectively. These analyses form the basis for the consideration of correspondence between the models, which is described in Section 6.5.

6.2 A Representation of DATAM Instances

A perception of DATAM instances are required in the analyses of data models. In order to relate to the structural representations of the relational and network models, a table form is used to represent DATAM associational instances. Different DATAM associations are represented by different table types.

6.2.1 EA-table type. Instances of the associations of an entity to its attributes are represented by an EA-table (Figure 12(a)). In this table, the first column represents a token instance set of an entity type and the other columns represent the instance sets of all the attribute types of that entity. Each row of the table indicates a particular association of the entity instance to its attribute instances. For example, the first row in Figure 12(a) represents an association of a token t_1 of the entity to particular attribute instances, e.g. v_{11} of attribute type A_1. Row two represents an association of the same token t_1 with another instance, v_{12}, of the attribute A_1.

6.2.2 R-table type. Tables of this type represent relationships. Each table (e.g. Figure 12(b)) consists of two columns containing token instances of the two participating object-types (entities or events).

6.2.3 Ev-table type. These tables represent events. Each table has columns representing the event tokens, tokens of the member objects, and the attribute instances of the event. These represent the existence of the event, the embedded relationships among the member objects (see Section 2.2.5), and the descriptions of the event. An Ev-table is illustrated in Figure 12(c).

6.3 Analysis of the Relational Model

In the relational model (Codd [8]), there is only one structure - the relation. The construction of relations based on functional dependency constraints results in

normalized relations (Codd [9], Bernstein [3]). These normalized relations have a correspondence with the tables in the DATAM representation of Section 6.2. This fact is used in the analysis of normalized relations where relations structurally similar to one of the DATAM tables are interpreted as representing the same concept as that of the table.

No direct correspondence however can be made between relational keys and DATAM tokens. The keys in the relational model uniquely identify tuples (rows of the tables) rather than db-entities. It is thus difficult to base a semantic analysis of a relational model on the internal structure of its relations as this reveals little of the represented concepts of the real world.

The stated functional dependencies of a relational model represent the perceived semantic constraints and they completely determine the construction of normalized relations. An analysis of a relational model can therefore be based on functional dependencies. Nevertheless, because functional dependencies are defined on representational rather than abstract structures, such interpretation is subject to ambiguity. This ambiguity can be resolved by relating each distinct dependency type to an abstract concept. The abstract constructs of DATAM can be used for this purpose, and the analysis of a relational model with DATAM is described here.

A set of dependencies (Bernstein [3]) for a relational

model is given in Figure 13(a). The set of normalized relations constructed from these dependencies as given by Bernstein is shown in Figure 13(b) and the DATAM interpretation of the model is presented in Figure 14.

6.3.1 Descriptor types. As stated, an analysis of a relational model here is based upon its dependencies. Consider the dependency, T : Stock# → Price, of Figure 13(a). The dependency represents the instance associations of Stock# and its Price. Here Price can be viewed as a descriptor of Stock#, that is, the Right Hand Side (RHS) of this dependency is a descriptor of the object on the Left Hand Side (LHS). This corresponds to a db-entity:db-attribute association so that the dependency T can be interpreted as the DATAM substructure T_d in Figure 14. T_d represents the Stock $^{n}:^{1}$ Price association where the entity Stock has Stock# as a token. The relational dependency W is similarly interpreted as the DATAM substructure W_d.

Although the RHS of a dependency, (such as T or W), is typically the descriptor of the LHS, this situation is not always so. For example, if the instance association for Stock#:Price is $1{:}n$ (instead of $n{:}1$), then T′ representing this dependency would be expressed as T′:Price → Stock#, where the descriptor (Price) is now on the LHS. To keep the descriptor on the RHS, a notational change for T′ to T′:Stock# ← Price could be made. This suggests a worthwhile notational rule for dependencies where the descriptor types are clearly identifiable. In the interpretation of Figure 13(a) it is assumed that the RHS always represents a descriptor.

6.3.2 Entity Type. In DATAM, a descriptor which is itself described is a db-entity. In the consideration of the dependencies U and W of Figure 13(a), City is a descriptor of Dept (from dependency U) and is itself described by Population (dependency W). Therefore City is to be viewed as a db-entity. Thus the Dept:City association is one between two entities, and is the db-relationship shown in the DATAM substructure, U_d, in Figure 14.

6.3.3 Event Type. Situations occur where either side of a dependency consists of more than one object (relational attribute). This combination represents an association among the constituent objects and is interpreted as a db-event. For example, the dependency S has the combination {Stock#,Dept#} which is interpreted as the db-event, Stock-in-Dept, involving the entities, Stock and Dept. The dependency S therefore represents the description of the event by the db-attribute, Quantity and is shown as substructure S_d in Figure 14.

6.3.4 Multivalued dependency. From the above analysis, functional dependencies are seen to represent association between objects and their descriptors. The definition of functional dependency (Codd [9]) allows only an *n:1* association. The inadequacy of this *n:1* associational restriction has recently motivated the introduction of the generalized *multivalued dependency*, where *m:n* associations can be specified (Fagin [11]). In the interpretation of general

dependencies, each DATAM association (*m:n* or *n:1*) follows that of the dependency.

6.4 Analysis of the Network Model

The network model has two basic structures - record type and set type. Network structures are generally represented by data structure diagrams (Bachman [2]).

6.4.1 Entity-attribute associations. A record type is used to represent the existence and description of a real world entity (Taylor and Frank [18]), so that, in value instances, it corresponds to the DATAM EA-table of Section 6.2. The keyfield of this record type corresponds to the token column of the EA-table and non-key fields correspond to attribute columns. Thus a record type can be interpreted as entity-attribute associations where a repeating group is viewed as an *n:m* entity-attribute instance association and a field within the record type is viewed as an *n:1* association.

Figure 15(a) shows a network model. The non-key fields for each of the record types are also indicated in the Figure. Record types are interpreted as db-entity : db-attribute association, so that the record type City corresponds to the DATAM substructure W_d in Figure 14. Similarly record type Stock corresponds to the DATAM substructure, T_d.

6.4.2 Relationship type. A set type represents the *1:n* association of record occurences of a record type to those of another. In value instances,therefore, a set

type corresponds to the DATAM R-table with an imposed *1:n* association restriction.

The set type, LOCATION, in the substructure U_n in Figure 15(a) represents the *1:n* association of City occurrences to Dept occurrences. This is interpreted as the db-relationship between the db-entities City and Dept in Figure 14.

6.4.3 Event type. Consider the substructure, S_n. Network structures of this form are ambiguous as they are used to represent two distinct concepts (Taylor and Frank [18]). In one case, the record type Stock-Dept-Link is seen as a db-entity so that the set types Held-in-Dept and Stock-Held are seen as two distinct db-relationships involving this entity.

On the other hand, the record type, Stock-Dept-Link, can be seen to facilitate a many-to-many association of the record types, Stock and Dept. In this case, the substructure, S_n, is interpreted as the DATAM substructure, S_d, containing a db-event.

To resolve the above ambiguity, in the analysis of a network model, the interpretation of network structures of the form of S_n has to be fixed, (see also Chen [6]). One method is to adopt the second approach discussed above. This involves the interpretation of record types that are owned as forming db-events. With this approach,however, the record type, Dept, in Figure 15(a) would become interpreted as a db-event rather than as a (intended) db-entity. To

overcome this, the association between Dept and City has to be represented through the substructure, X_n, shown in Figure 15(b) where a dummy record type is introduced to facilitate the association. With this change, the relationship between the entities, Dept and City becomes embedded in the db-event, Dept-City-Link. Therefore, in this method, there is no network structure than can be directly interpreted as a db-relationship.

6.5 Correspondence of Data Models

The transformation of a data model to another based on their structural constructs is semantically intuitive (Biller and Neuhold [4]) and therefore provides no guarantee of the validity of the correspondence where the interpretation of particular structures may be ambiguous. To ensure a valid correspondence, a semantic reference is used to impose an interpretation on the models with respect to which the correspondence is to be defined. DATAM provides a basis for such an interpretation.

This Section discusses the correspondence between the network and relational models in terms of the DATAM interpretations presented in Sections 6.3 and 6.4. In the interpretation of the network model presented in Section 6.4, a discipline was imposed where dummy record types are introduced in order to preserve the entity role of some record types. A consequence of this is that db-relationships have no distinct network representation, and therefore transformations between

relational and network models may not be reversible.

6.5.1 Relational to network transformation. This involves the translation of any given set of relational dependencies into an equivalent network structure. Consider, for example, the dependencies in Figure 13(a). A DATAM interpretation of this relational model is given in Figure 14. The network representation of this DATAM model is that of Figure 15(b). Therefore Figure 15(b) is the corresponding network model for the relational model of Figure 13(a).

6.5.2 Network to relational transformation. The reverse transformation of the network model of Figure 15(b) does not result in the relational dependencies of Figure 13(a). Specifically, the substructure, X_n, cannot be interpreted as the DATAM substructure U_d in Figure 14. The DATAM interpretation of X_n contains a db-event rather than a db-relationship. This DATAM interpretation, being different from that of Figure 14, will not produce the set of dependencies of Figure 13(a).

A reversible transformation is possible if the relational model involved does not contain dependencies which become db-relationships in the DATAM interpretation.

7. SUMMARY AND CONCLUSION

A data abstraction model has been presented for the modelling and representation of data for database systems. Concept abstractions are defined for the representation of the enterprise view and value abstractions are then defined for the representation of instances in the database.

These definitions mean that, in DATAM, the concept abstraction and value abstraction domains are clearly divided into constructs which represent types of objects and their associations, and abstractions which represent the values that the instances of the construct types can have.

The expressiveness of DATAM is illustrated by examples in this paper. Its capabilities in the construction of subviews, and in evolution to reflect changes in perception are described. The use of DATAM in the interpretation of data models is also presented.

ACKNOWLEDGMENT

One of us, M.M. Ridwan, acknowledges the support of a New Zealand University Grants Committee Postgraduate Scholarship.

REFERENCES

1. ANSI/X3/SPARC, Study Group on Data Base Management Systems: Interim Report, ANSI, 75-02-08. In *FDT, ACM-SIGMOD 7*, 2 (1975).

2. BACHMAN, C.W. Data structure diagrams. *Database 1*, 2 (1969), 4-10.

3. BERNSTEIN, P.A. Normalization and functional dependencies in the relational data base model. Technical Report CSRG-60, Computer Systems Research Group, University of Toronto, Oct. 1975.

4. BILLER, H., and NEUHOLD, E.J. Semantics of data bases: the structure of data models. Technical Report 03/76, Institut für Informatik, Universität Stuttgart, 1976.

5. BILLER, H., and NEUHOLD, E.J. Concepts for the conceptual schema. In *Architecture and Models in Data Base Management Systems*, G.M. Nijssen, Ed., North-Holland Pub. Co., Amsterdam, 1977, pp. 1-30.

6. CHEN, P. P-S. The entity-relationship model - toward a unified view of data. *ACM Tran. on Database Systems 1*, 1 (March 1976), 9-36.

7. CHEN, P. P-S. The entity-relationship model - a basis for the enterprise view of data. Proc., AFIPS 1977 NCC, Vol. 46, AFIPS Press, Montvale, N.J., pp.77-84.

8. CODD, E.F. A relational model of data for large shared data banks. *Comm. ACM 13*, 6 (June 1970), 377-387.

9. CODD, E.F. Further normalization of the data base relational model. In *Courant Computer Science Symposia 6: Data Base Systems*, R. Rustin, Ed., Prentice-Hall, Englewood Cliffs, N.J., 1972.

10. DATE, C.J. *An Introduction to Database Systems.* Addison-Wesley, Reading, Mass., 1975.

11. FAGIN, R. Multivalued dependencies and a new normal form for relational databases. *ACM Tran. on Database Systems 2*, 3 (Sept. 1977), 262-278.

12. FRY, J.P. and SIBLEY, E.H. Evolution of data-base management systems. *Computing Surveys 8*, 1 (March 1976), 7-42.

13. KERSCHBERG, L., KLUG, A., and TSICHRITZIS, D. A taxonomy of data models. In *Systems for Large Data Bases*, P.C. Lockemann and E.J. Neuhold, Eds., North-Holland Pub. Co., Amsterdam, 1976, pp.43-63.

14. NAVATHE, S.B. and FRY, J.P. Restructuring for large databases: three levels of abstraction. *ACM Tran. on Database Systems 1*, 2 (June 1976), 138-158.

15. SCHMID, H.A. and SWENSON, J.R. On the semantics of the relational data model. Proc. ACM-SIGMOD 1975, Conference, San Jose, Calif., May 1975, pp. 211-223.

16. SIBLEY, E.H. and KERSCHBERG, L. Data architecture and data model considerations. Proc., AFIPS 1977 NCC, Vol. 46, AFIPS Press, Montvale, N.J., pp. 85-96.

17. SWARTOUT, D.E., DEPPE, M.E., and FRY, J.P. Operational software for restructuring network databases. Proc., AFIPS 1977 NCC, Vol. 46, AFIPS Press, Montvale, N.J., pp.499-508.

18. TAYLOR, R.W. and FRANK, R.L. Codasyl data-base management systems. *Computing Surveys 8*, 1 (March 1976), 67-103.

19. TSICHRITZIS, D., and LOCHOVSKY, F. Views on data. Proc. Second SHARE Working Conference, Montreal, Canada, April 26-30, 1976, North-Holland Pub. Co., Amsterdam, 1977, pp.51-65.

FIGURE CAPTIONS

Fig. 1. DATAM diagrammatic representations
(a) concept abstraction types and their associations
(b) representation of value abstractions

Fig. 2. Example of DATAM diagram

Fig. 3. Entity associations
(a) 4-way event
(b), (c) other representations involving the entities

Fig. 4. Instances for the relational example in Section 2.3.1

Fig. 5. An entity-relationship model and instances for the example in Section 2.3.1

Fig. 6. A DATAM model and instances for the example in Section 2.3.1

Fig. 7. A DATAM diagram for the supply model of Section 2.3.2

Fig. 8. Example of value abstraction

Fig. 9. Dependency objects
(a) a dependency-relationship
(b) dependency-event

Fig. 10. Evolution of a DATAM model

Fig. 11. The sub-entity type

Fig. 12. DATAM instance representations
(a) entity-attribute, EA-table type
(b) relationship, R-table type
(c) event, Ev-table type

Fig. 13. A relational model for analysis
(a) relational dependencies
(b) normalised relations

Fig. 14. A DATAM interpretation

Fig. 15. A network model for analysis
(a) data structure diagram
(b) modified diagram

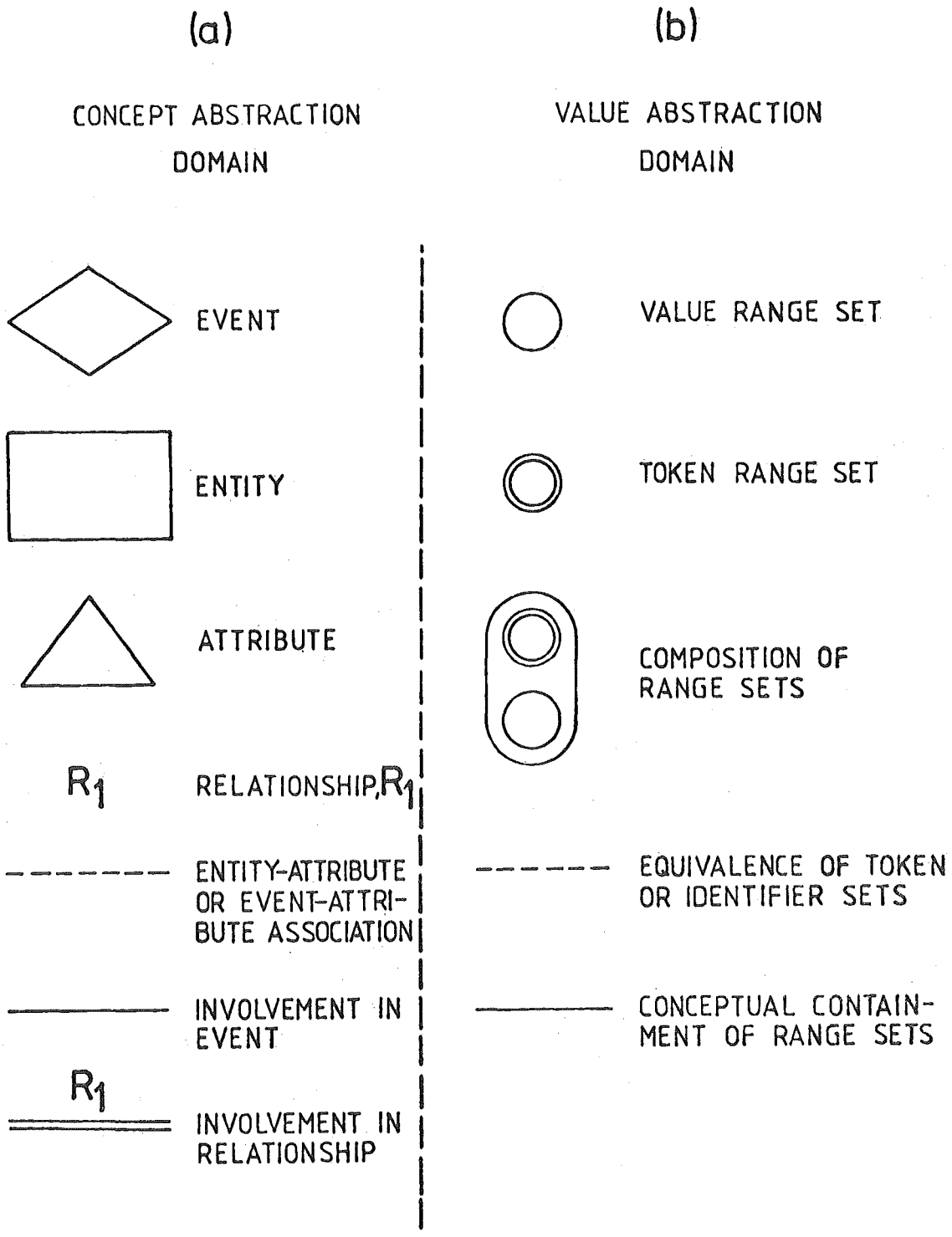


FIGURE 1. DATAM diagrammatic representations

(a) concept abstraction types and their associations
(b) representation of value abstractions

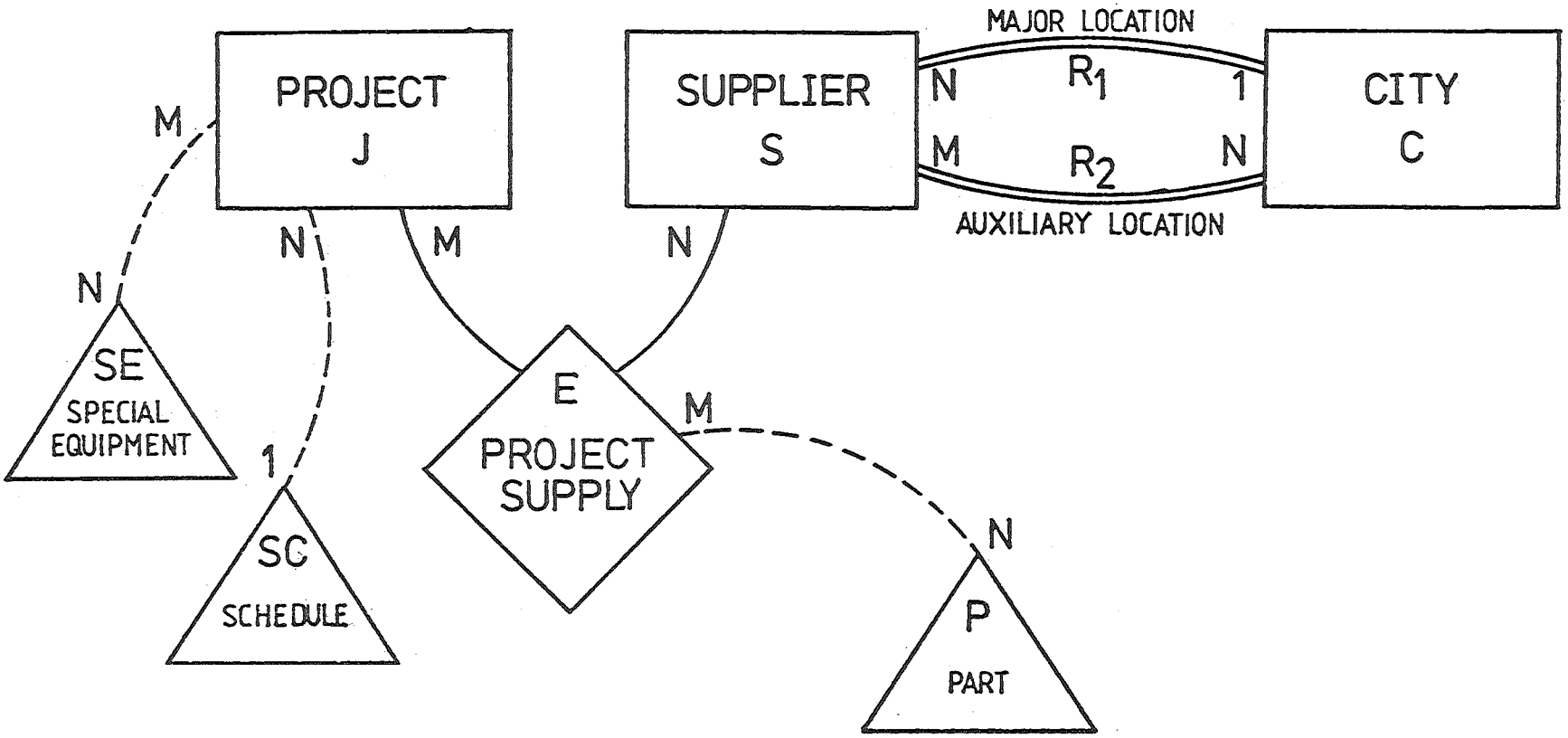


FIGURE 2. Example of DATAM diagram

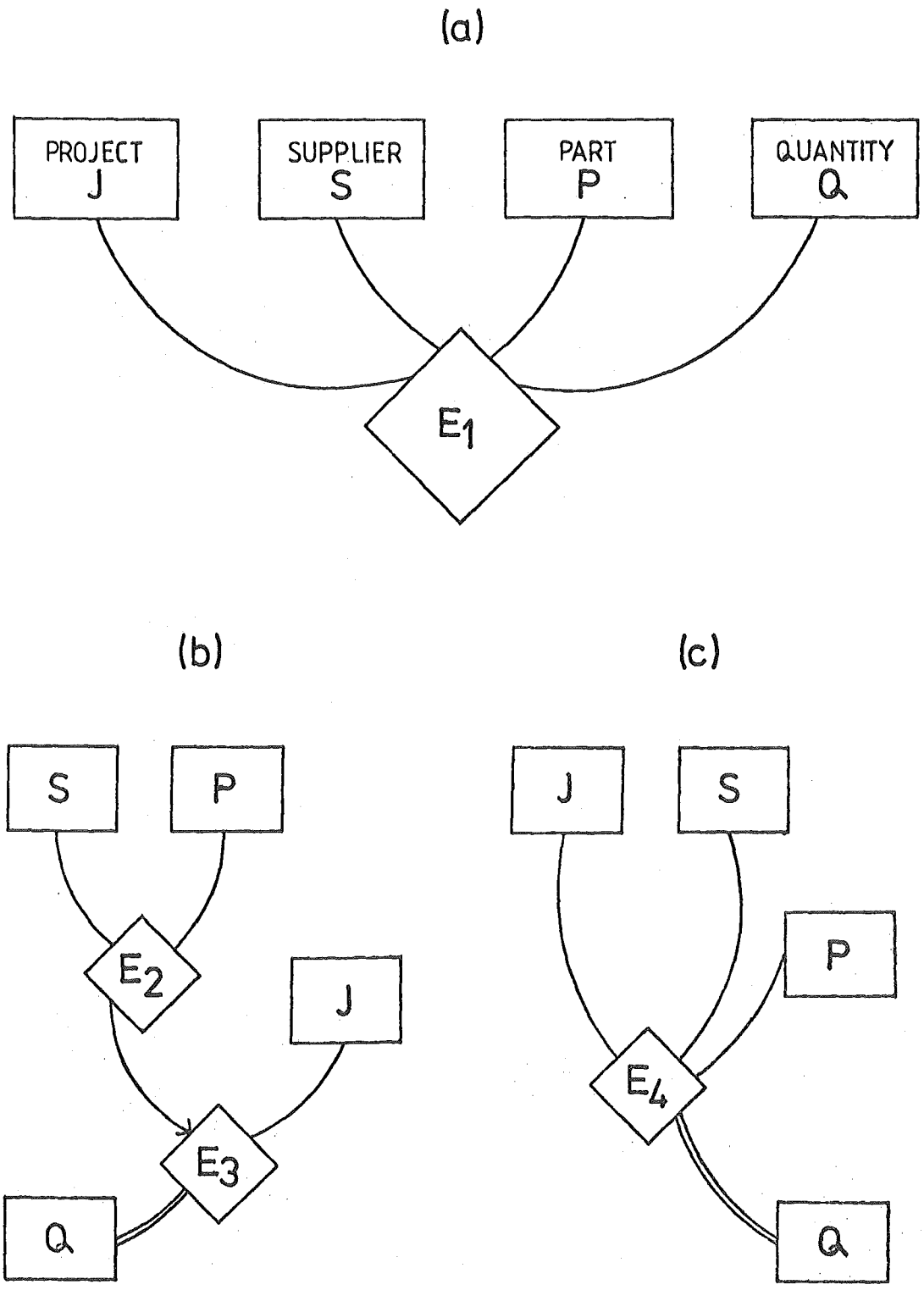


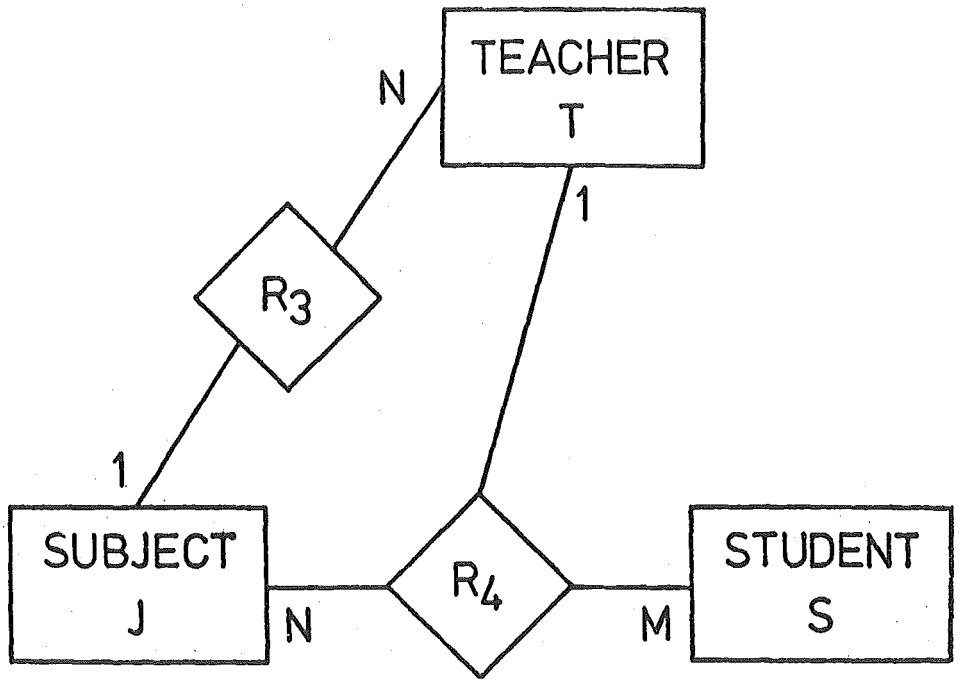
FIGURE 3. Entity associations

(a) 4-way event
(b), (c) other representations involving the entities

R1 (S , T)	
S1	T1
S2	T2
S2	T3

R2 (T , J)	
T1	J1
T2	J1
T3	J2

FIGURE 4. Instances for the relational example in Section 2.3.1



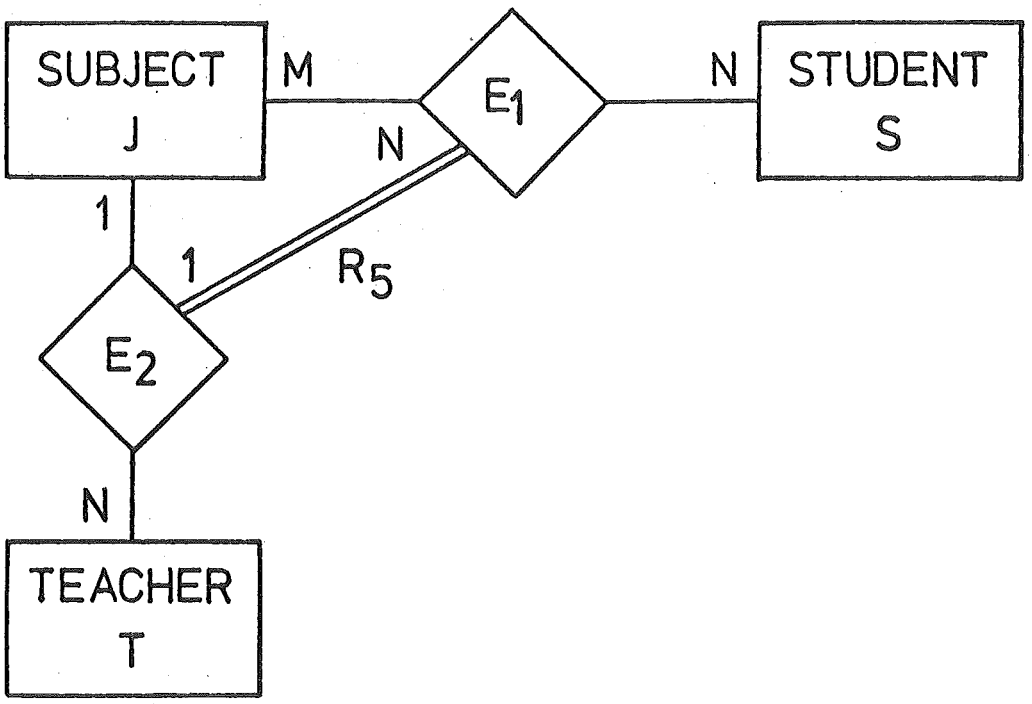
R3 (J , T)

J	T
J1	T1
J1	T2
J2	T3

R4 (J , S , T)

J	S	T
J1	S1	T1
J1	S2	T2
J2	S2	T3
J2	S3	T1

FIGURE 5. An entity-relationship model and instances for the example in Section 2.3.1



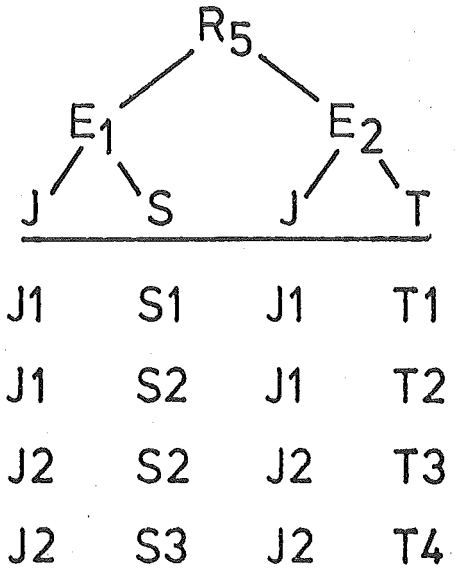


FIGURE 6. A DATAM model and instances for the example in Section 2.3.1

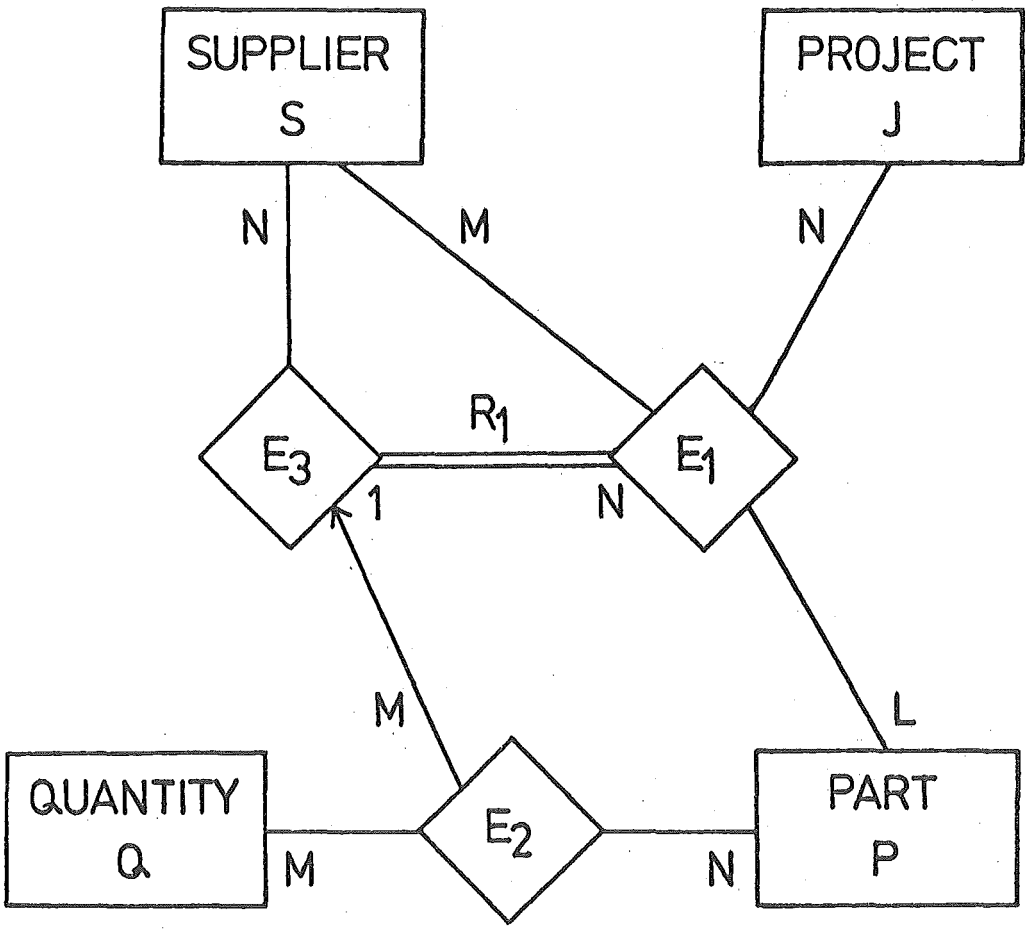


FIGURE 7. A DATAM diagram for the supply model of Section 2.3.2

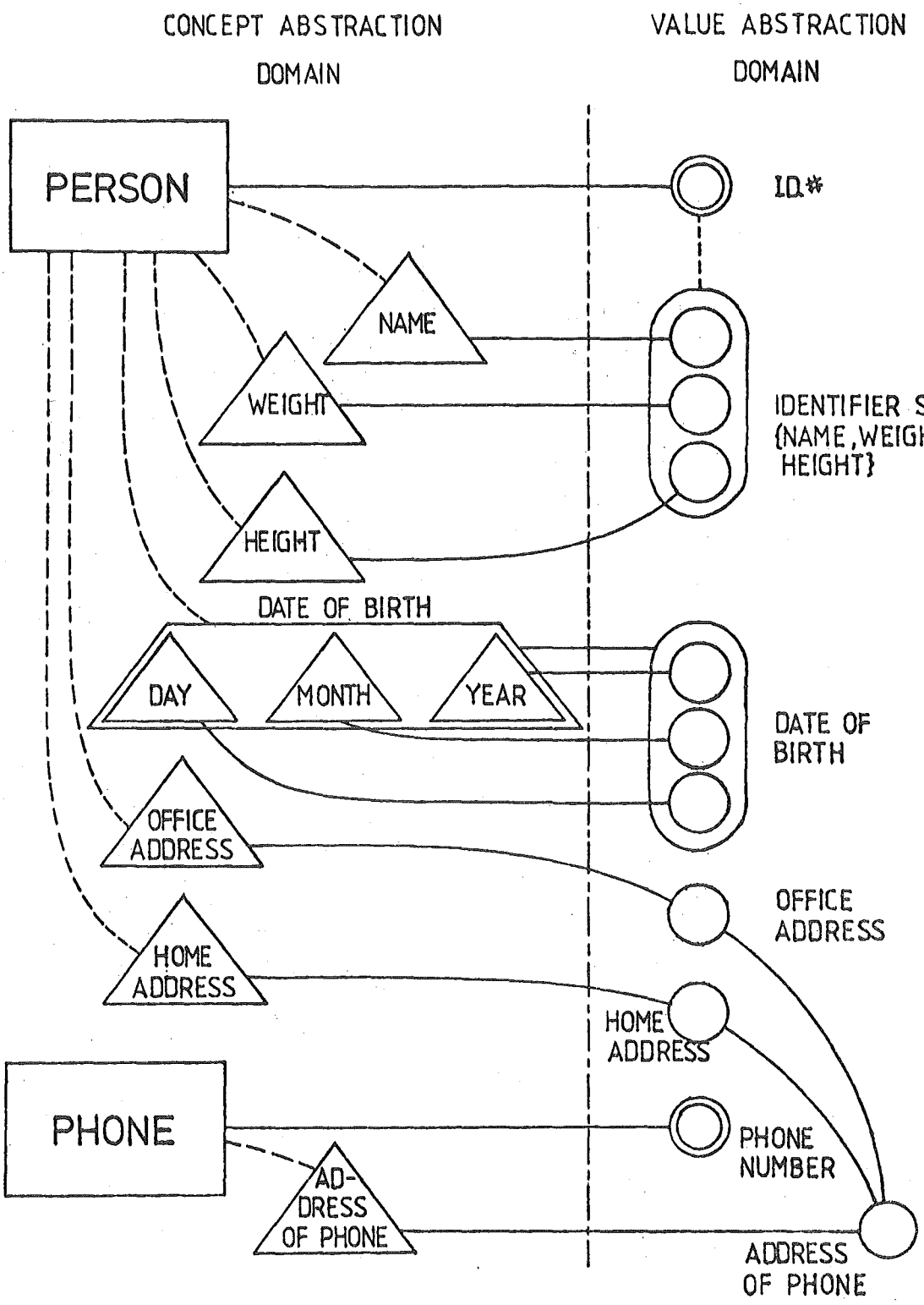


FIGURE 8. Example of value abstraction

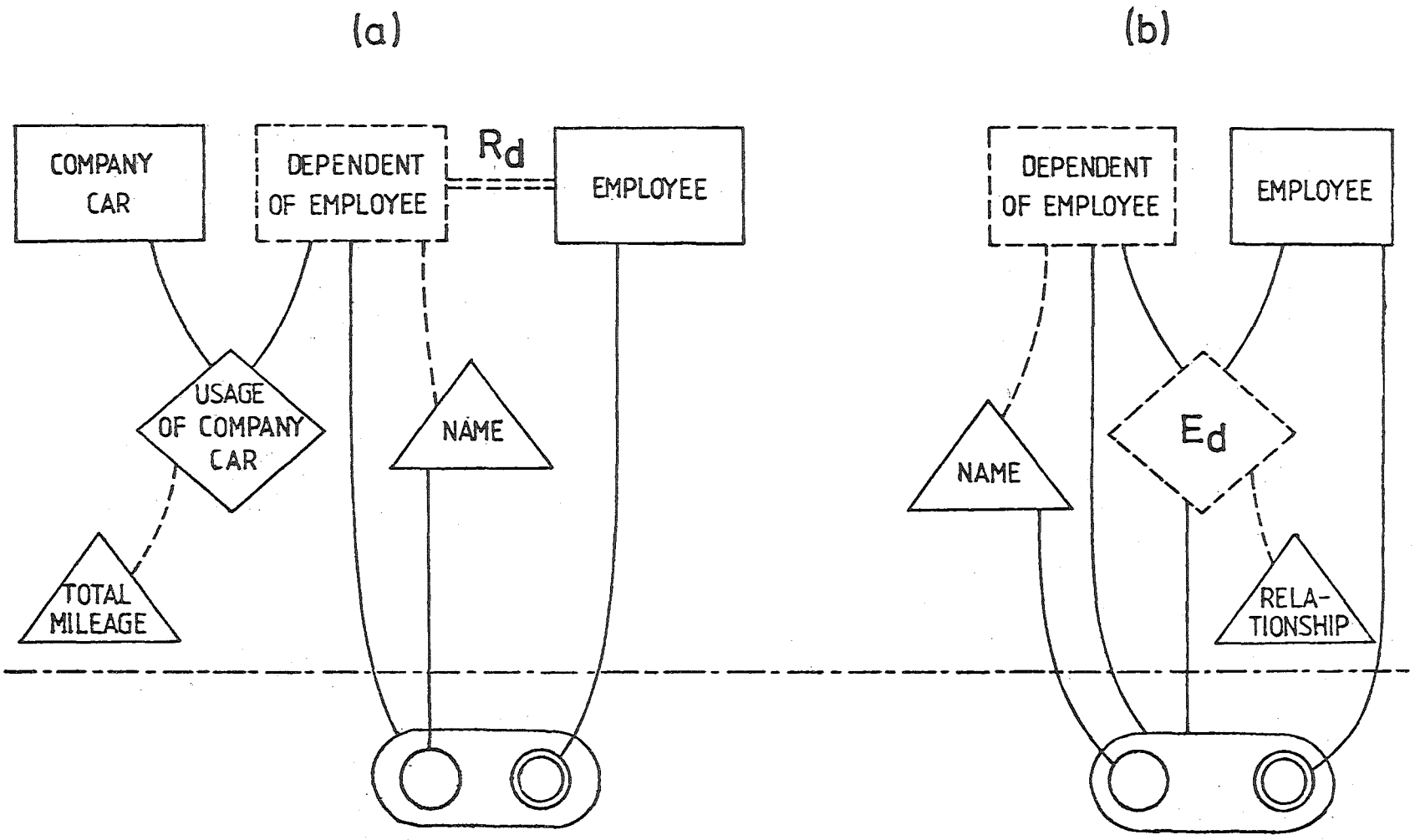


FIGURE 9. Dependency objects

(a) a dependency-relationship
(b) dependency-event

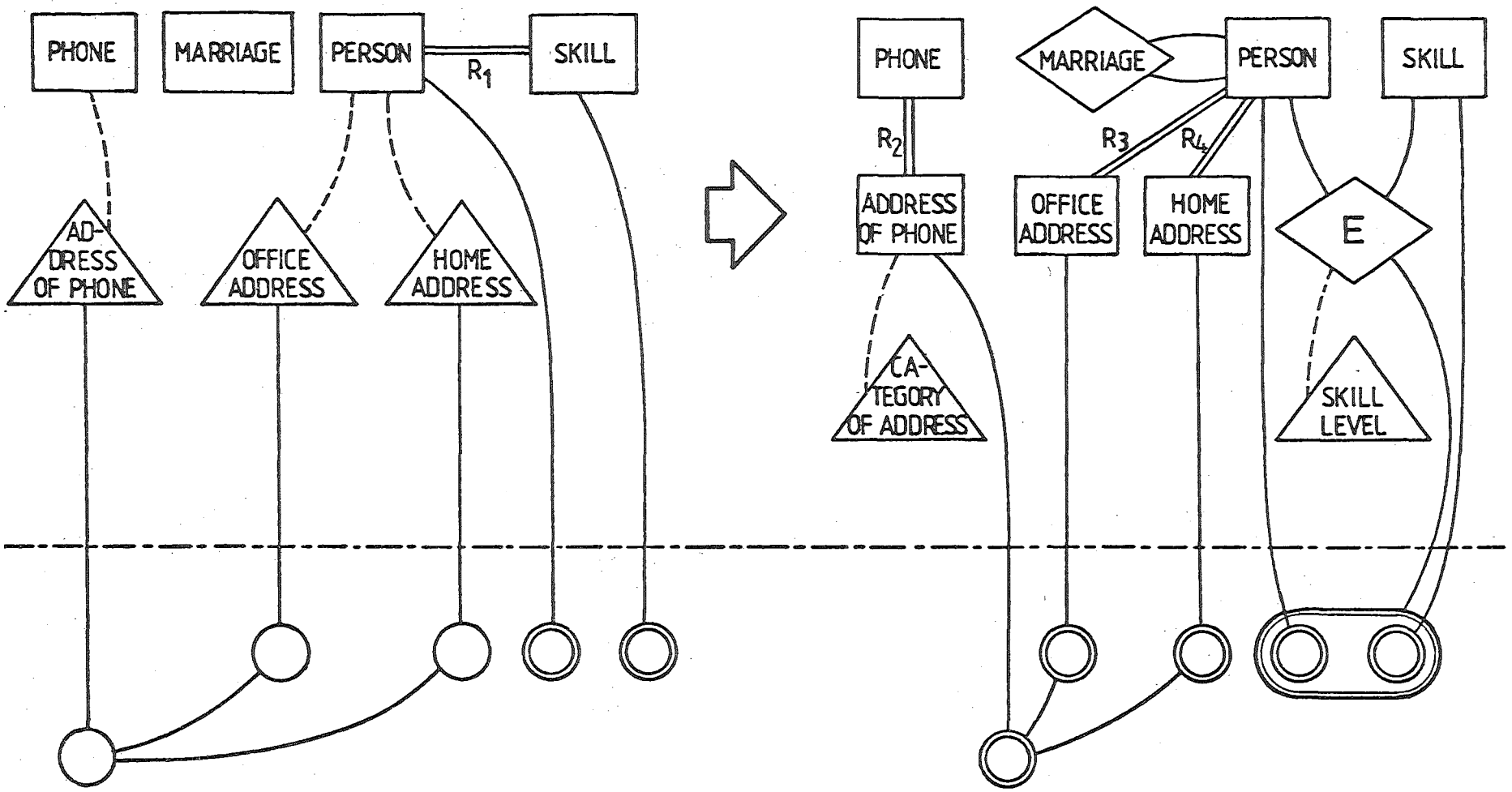


FIGURE 10. Evolution of a DATAM model

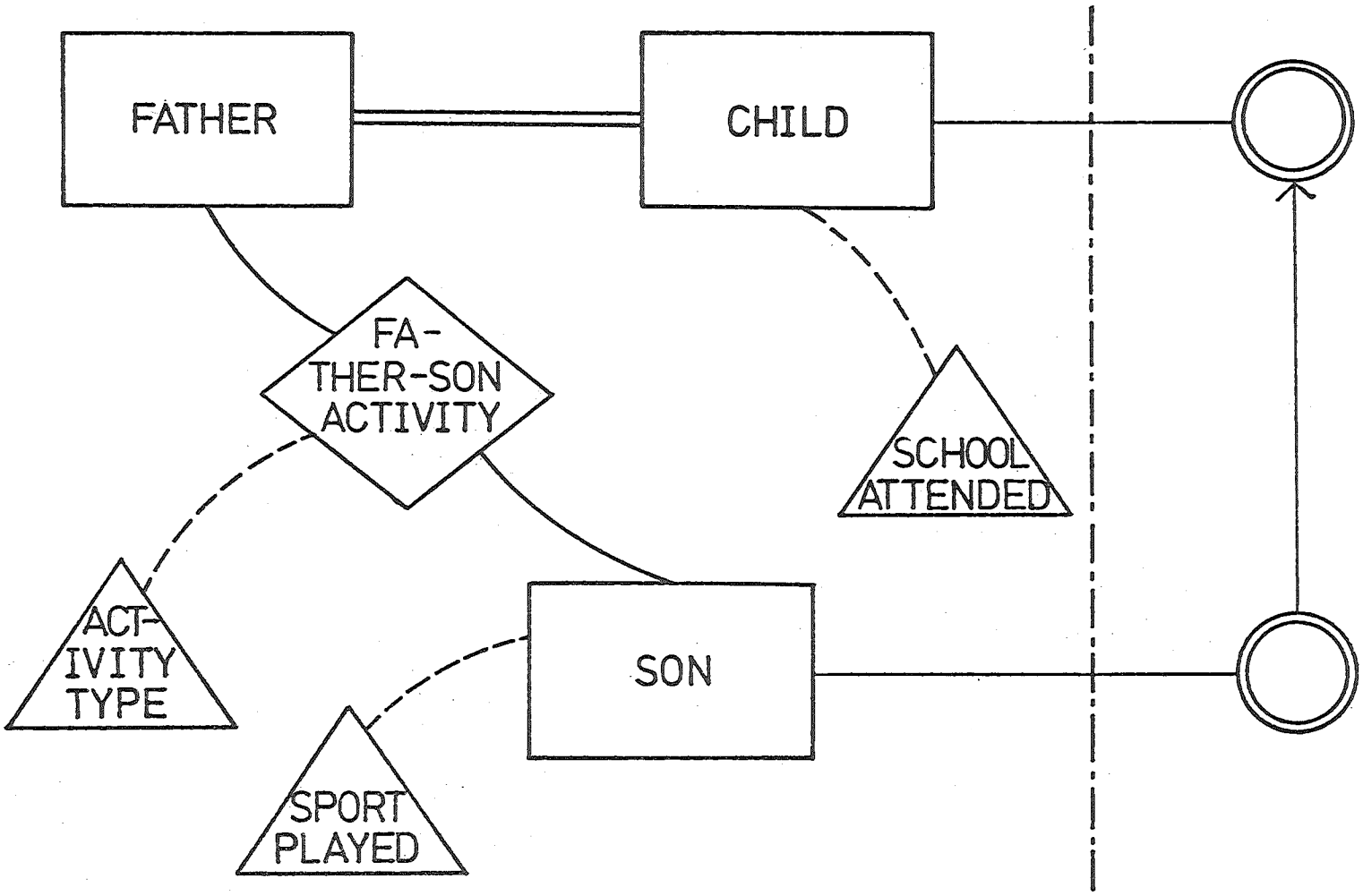


FIGURE 11. The sub-entity type

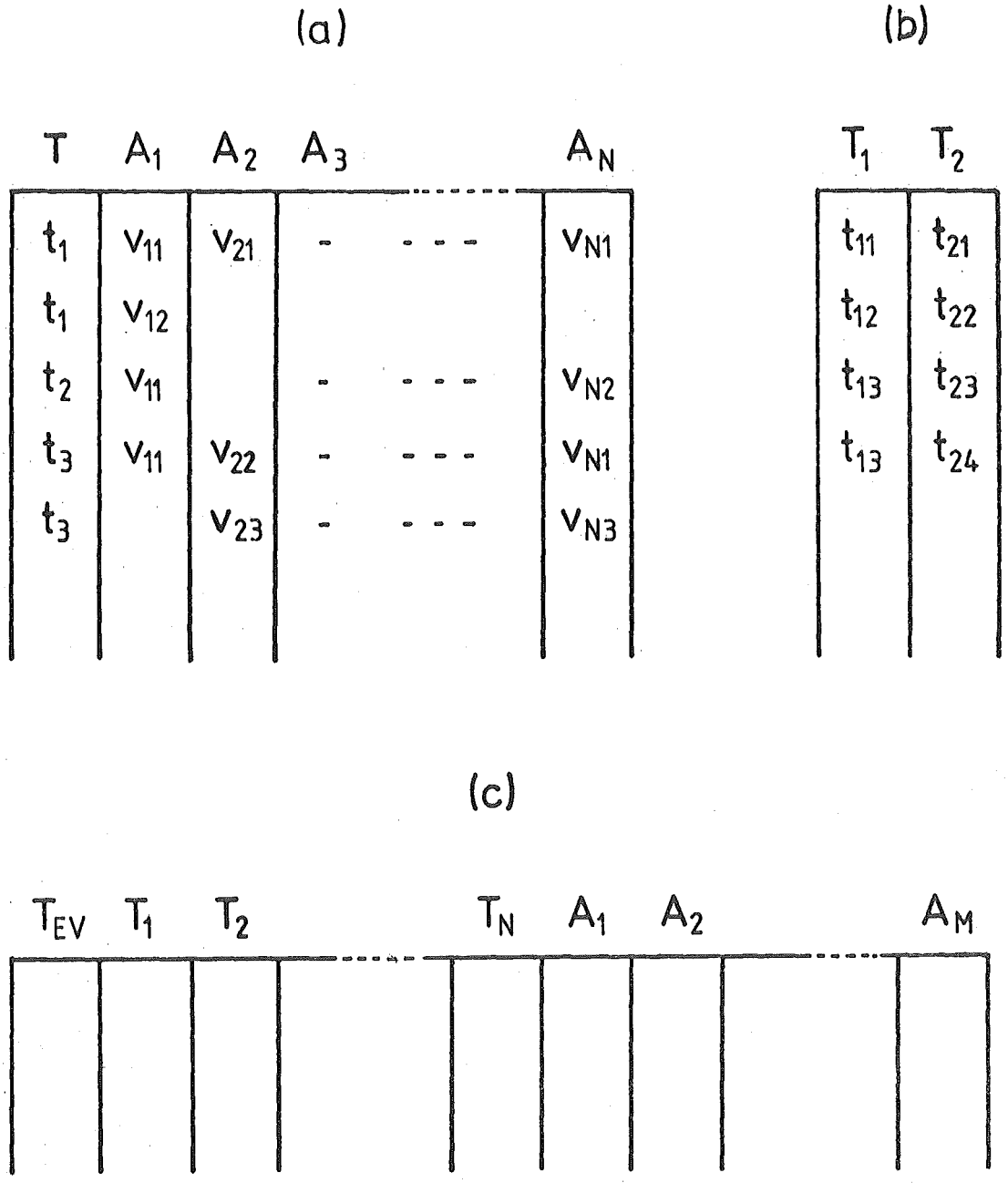


FIGURE 12. DATAM instance representations

(a) entity-attribute, EA-table type
(b) relationship, R-table type
(c) event, Ev-table type

(a)

S: {STOCK#, DEPT#} → QUANTITY

T: STOCK# → PRICE

U: DEPT# → CITY

W: CITY → POPULATION

(b)

S (STOCK#, DEPT#, QUANTITY)

T (STOCK#, PRICE)

U (DEPT#, CITY)

W (CITY, POPULATION)

FIGURE 13. A relational model for analysis
(a) relational dependencies
(b) normalised relations

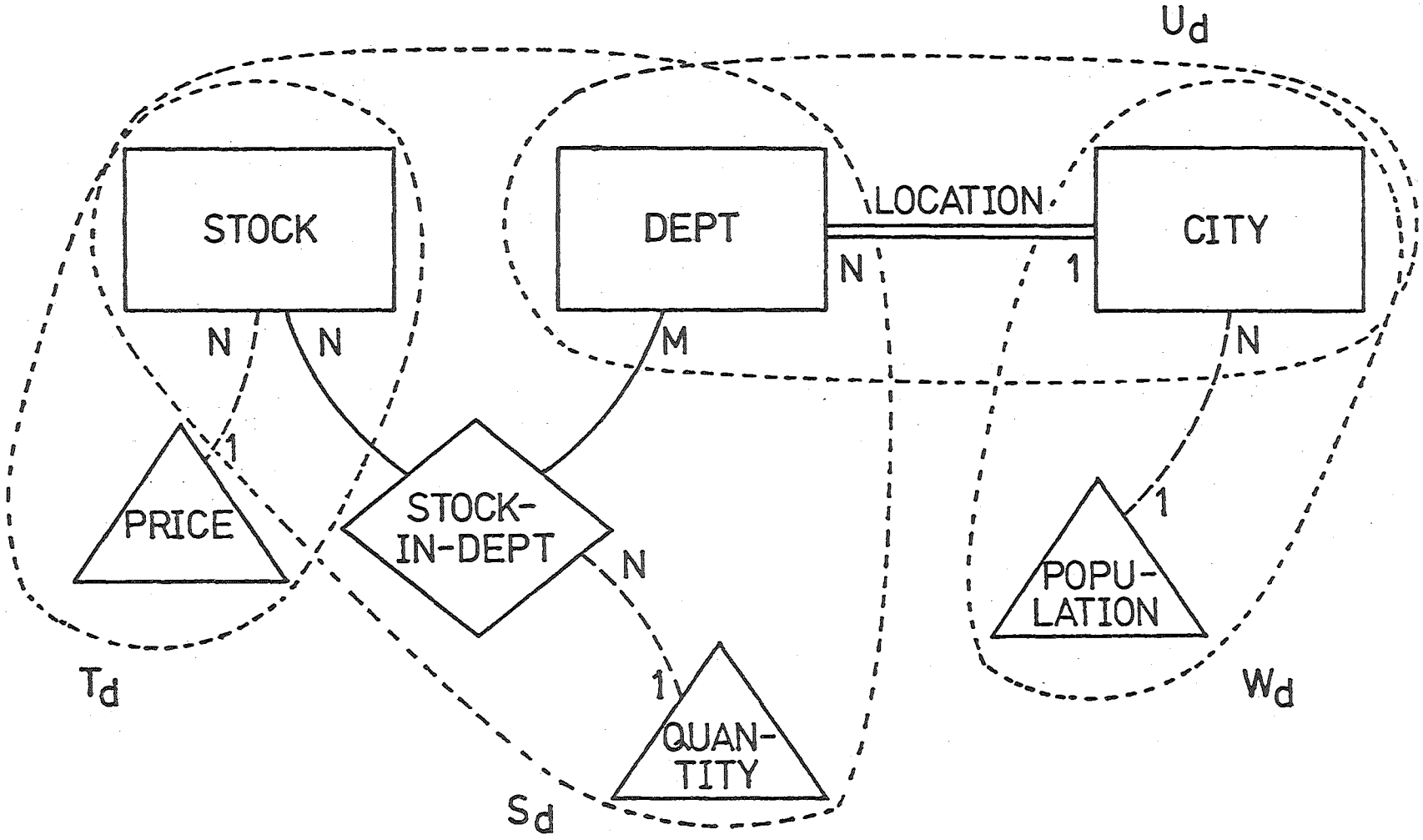


FIGURE 14. A DATAM interpretation

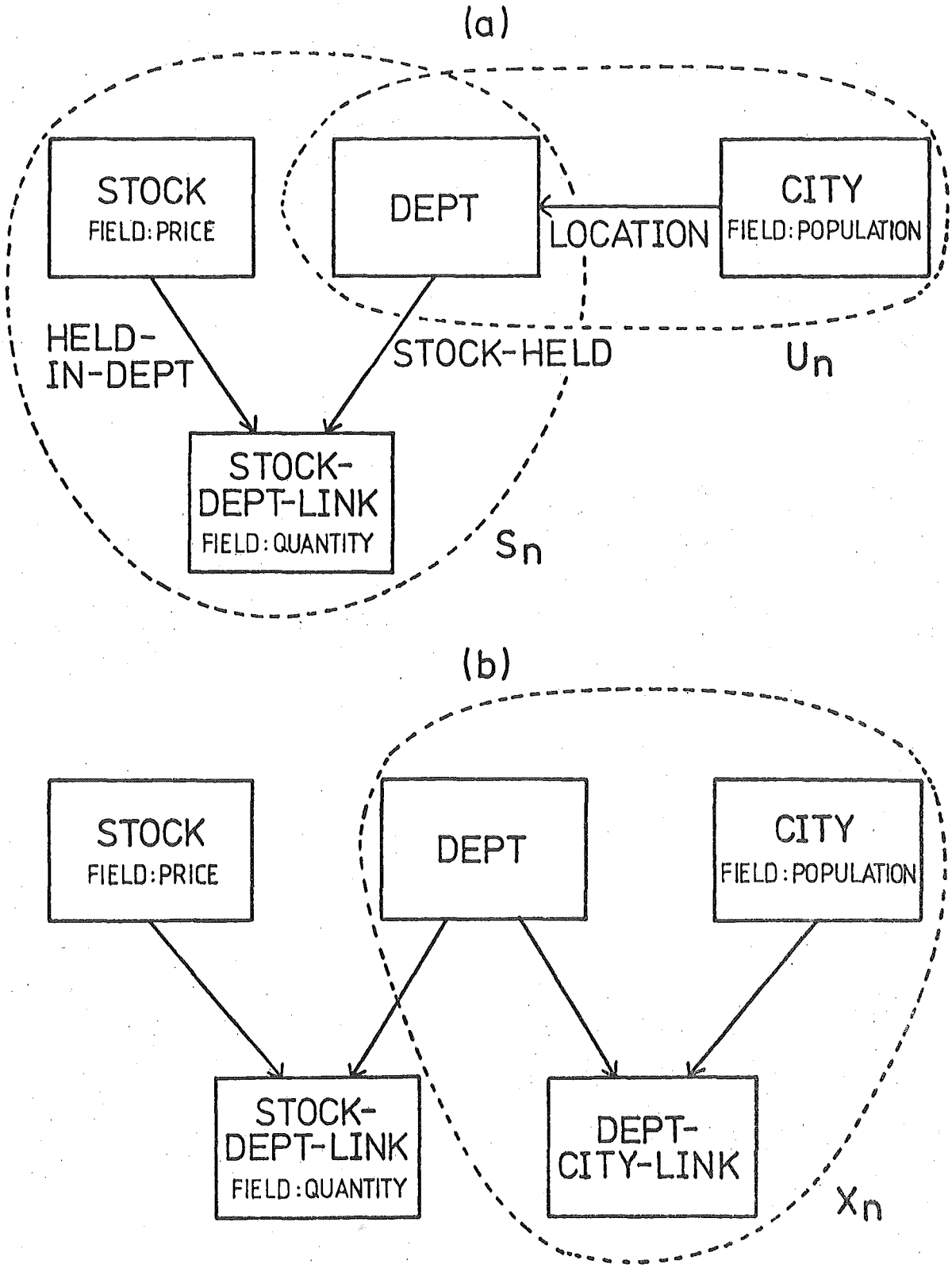


FIGURE 15. A network model for analysis

(a) data structure diagram
(b) modified diagram